Weighted Model Counting with Conditional Weights for Bayesian Networks

Abstract

Weighted model counting (WMC) has emerged as the unifying inference mechanism across many (probabilistic) domains. Encoding an inference problem as an instance of WMC typically necessitates adding extra literals and clauses. This is partly so because the predominant definition of WMC assigns weights to models based on weights on literals, and this severely restricts what probability distributions can be represented. We develop a measure-theoretic perspective on WMC and propose a way to encode conditional weights on literals analogously to conditional probabilities. This representation can be as succinct as standard WMC with weights on literals but can also expand as needed to represent probability distributions with less structure. To demonstrate the performance benefits of conditional weights over the addition of extra literals, we develop a new WMC encoding for Bayesian networks and adapt a state-of-the-art WMC algorithm ADDMC to the new format. Our experiments show that the new encoding results in significantly faster inference on most benchmark instances. Furthermore, the same idea can be adapted to other WMC algorithms and other problem domains.

1 Introduction

Weighted model counting (WMC), i.e., an extension of model counting (#SAT) that assigns a weight to every model [Sang *et al.*, 2005], has emerged as one of the most dominant and competitive approaches for handling inference tasks in a wide range of formalisms including Bayesian networks [Sang *et al.*, 2005] and probabilistic programs [Fierens *et al.*, 2015]. Furthermore, by generalising the notion of weights to an arbitrary semiring, a range of other problems are also captured [Kimmig *et al.*, 2017]. Exact WMC solvers typically rely on either knowledge compilation [Lagniez and Marquis, 2017] or exhaustive DPLL search [Sang *et al.*, 2005].

The most well-studied version of WMC assigns weights to models based on weights on literals, i.e., the weight of a model is the product of the weights of all literals in it. This simplification is motivated by the fact that the number of models scales exponentially with the number of atoms, so listing the weight of every model is intractable. However, this also severely restricts what probability distributions can be represented. A common way to overcome this limitation is by adding more literals. While we show that this is always possible, we also demonstrate that it can be significantly more efficient to encode weights in a more flexible format instead.

After briefly reviewing the background in Section 2, in Section 3 we describe three equivalent perspectives on the subject based on logic, set theory, and Boolean algebras. Furthermore, we describe the space of functions on Boolean algebras and various operations on those functions. Section 4 introduces WMC as the problem of computing the value of a measure on a Boolean algebra. We show that not all measures can be represented using literal-based WMC, but all Boolean algebras can be extended to make any measure representable in such a manner.

This new measure-theoretic perspective allows us to not only encode any discrete probability distribution but also improve inference speed. In Section 5, we demonstrate this by developing a new WMC encoding for Bayesian networks that uses *conditional weights* on literals (in the spirit of conditional probabilities) that have literal-based WMC as a special case. We prove the correctness of the encoding and show how a state-of-the-art WMC solver ADDMC [Dudek *et al.*, 2020] can be adapted to the new format. ADDMC is a recently-proposed algorithm for WMC based on manipulating functions on Boolean algebras using an efficient representation for such functions known as algebraic decision diagrams (ADDs) [Bahar *et al.*, 1997]. Finally, in Section 6, we show how the new encoding results in faster inference on the vast majority of benchmark instances, often by one or two orders of magnitude. We explain the performance benefits by showing how our encoding has asymptotically fewer variables and ADDs.

2 Related Work

WMC for Bayesian networks. Hitherto, four techniques have been proposed for encoding Bayesian networks into instances of WMC. We will identify them based on the initials of authors as well as publications years: `d02` [Darwiche, 2002], `sbk05` [Sang *et al.*, 2005], `cd05` [Chavira and Darwiche, 2005], and `cd06` [Chavira and Darwiche, 2006]. Below we summarise the observed performance differences among them. Sang *et al.* [2005] claim that `sbk05` is

a smaller encoding than d02 with respect to both the number of clauses and the number of variables but provide no experimental comparison. Chavira and Darwiche [2005] compare cd05 with d02 by measuring the time it takes to compile either encoding into an arithmetic circuit (but do not measure inference time). The more recent encoding cd05 always compiles faster and results in a smaller arithmetic circuit (as measured by the number of edges). In their subsequent paper, the same authors perform two sets of experiments (that are relevant to this summary) [Chavira and Darwiche, 2006]. First, they compile cd05 and cd06 encodings into d-DNNF (i.e., deterministic decomposable negation normal form [Darwiche, 2001]), measuring both compilation time and numbers of edges in the d-DNNF diagram. The results are mostly in favour of cd06. Second, they compare the inference time of sbk05 run with Cachet [Sang *et al.*, 2004] with the compile times of cd05 and cd06, but only on five (types of) instances. In these experiments, cd06 is always faster than cd05, while the comparison with sbk05 is mixed. Sometimes cd06 is orders of magnitude faster than sbk05, sometimes slightly slower. The performance difference between sbk05 and cd05 is even harder to judge: sbk05 is better on three out of five instances and worse on the remaining two. Based on this description, one would expect cd06 to be faster than both cd05 and sbk05, both of which should be faster than d02. The experiments in Section 6, however, strongly disagree with this prediction, showing that the quality of an encoding depends strongly on the underlying search algorithm or compilation technique.

3 Boolean Algebras, Power Sets, and Propositional Logic

In this section, we give a brief introduction to two alternative ways to think about logical constructs such as models and formulas. Let us consider a simple example of a propositional logic $\mathcal{L}$ with only two atoms a and b, and let $U = \{a, b\}$. Then 2^U, the power set of U, is the set of all models of $\mathcal{L}$, and 2^{2^U} is the set of all formulas. These sets can also be represented as Boolean algebras (e.g., using the syntax $(2^{2^U}, \wedge, \vee, \neg, \bot, \top)$) with a partial order $\leq$ that corresponds to set inclusion $\subseteq$—see Table 1 for examples of how various elements can be represented in both notations. Most importantly, note that the word *atom* has completely different meanings in logic and in Boolean algebras. An atom in $\mathcal{L}$ is an atomic formula, i.e., an element of U, whereas an atom in a Boolean algebra is (in set-theoretic terms) a singleton set. For instance, an atom in 2^{2^U} corresponds to a model of $\mathcal{L}$, i.e., an element of 2^U. Unless referring specifically to a logic, we will use the algebraic definition of an atom and refer to logical atoms as *variables*. In the rest of the paper, for any set U, we will use set-theoretic notation for 2^U and Boolean-algebraic notation for 2^{2^U}, except for (Boolean) atoms in 2^{2^U} that are denoted as $\{x\}$ for some model $x \in 2^U$.

3.1 The Space of Functions on Boolean Algebras

We also consider the space of all functions from any Boolean algebra to $\mathbb{R}_{\geq 0}$ together with some operations on those functions. They will be instrumental in defining WMC as a measure in Section 4 and can be efficiently represented using ADDs. Furthermore, all of the operations are supported by CUDD package that is used by ADDMC for ADD manipulation [Dudek *et al.*, 2020]. The definitions of multiplication and projection are as defined by Dudek *et al*. [2020], while others are new.

Definition 1 (Operations on functions)**.** *Let $\alpha\colon 2^X \to \mathbb{R}_{\geq 0}$ and $\beta\colon 2^Y \to \mathbb{R}_{\geq 0}$ be functions, $p \in \mathbb{R}_{\geq 0}$, and $x \in X$. We define the following operations:*

Addition: *$\alpha + \beta\colon 2^{X \cup Y} \to \mathbb{R}_{\geq 0}$ is such that $(\alpha + \beta)(T) = \alpha(T \cap X) + \beta(T \cap Y)$ for all $T \in 2^{X \cup Y}$.*

Multiplication: *$\alpha \cdot \beta\colon 2^{X \cup Y} \to \mathbb{R}_{\geq 0}$ is such that $(\alpha \cdot \beta)(T) = \alpha(T \cap X) \cdot \beta(T \cap Y)$ for all $T \in 2^{X \cup Y}$.*

Scalar multiplication: *$p\alpha\colon 2^X \to \mathbb{R}_{\geq 0}$ is such that $(p\alpha)(T) = p \cdot \alpha(T)$ for all $T \in 2^X$.*

Complement: *$\overline{\alpha}\colon 2^X \to \mathbb{R}_{\geq 0}$ is such that $\overline{\alpha}(T) = 1 - \alpha(T)$ for all $T \in 2^X$.*

Projection: *$\exists_x \alpha\colon 2^{X \setminus \{x\}} \to \mathbb{R}_{\geq 0}$ is such that $(\exists_x \alpha)(T) = \alpha(T) + \alpha(T \cup \{x\})$ for all $T \in 2^{X \setminus \{x\}}$. For any $Z = \{z_1, \ldots, z_n\} \subseteq X$, we write $\exists_Z$ to mean $\exists_{z_1} \ldots \exists_{z_n}$.*

Note that if U is any set, then $\mathcal{V} = \{\alpha\colon 2^X \to \mathbb{R}_{\geq 0} \mid X \subseteq U\}$ is a semi-vector space with three additional operations: (non-scalar) multiplication, complement, and projection. Specifically, note that both addition and multiplication are both associative and commutative.

We end the discussion on function spaces by defining several special functions: unit $1\colon 2^\emptyset \to \mathbb{R}_{\geq 0}$ defined as $1(\emptyset) = 1$, zero $0\colon 2^\emptyset \to \mathbb{R}_{\geq 0}$ defined as $0(\emptyset) = 0$, and function $[a]\colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ defined as $[a](\emptyset) = 0$, $[a](\{a\}) = 1$ for any a. Henceforth, for any function $\alpha\colon 2^X \to \mathbb{R}_{\geq 0}$ and any set T, we will write $\alpha(T)$ to mean $\alpha(T \cap X)$.

4 WMC as a Measure on a Boolean Algebra

In this section, we introduce an alternative definition of WMC and demonstrate how it relates to the standard one. Let U be a set. A *measure* is a function $\mu\colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ such that $\mu(\bot) = 0$, and $\mu(a \vee b) = \mu(a) + \mu(b)$ for all $a, b \in 2^{2^U}$ whenever $a \wedge b = \bot$. A *weight function* is a function $\nu\colon 2^U \to \mathbb{R}_{\geq 0}$. A weight function is *factored* if $\nu = \prod_{x \in U} \nu_x$ for some functions $\nu_x\colon 2^{\{x\}} \to \mathbb{R}_{\geq 0}$, $x \in U$. We say that a weight function $\nu\colon 2^U \to \mathbb{R}_{\geq 0}$ *induces* a measure $\mu_\nu\colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ if $\mu_\nu(x) = \sum_{\{u\} \leq x} \nu(u)$.

Theorem 1. *The function μ_ν is a measure.*

Finally, a measure $\mu\colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ is *factorable* if there exists a factored weight function $\nu\colon 2^U \to \mathbb{R}_{\geq 0}$ that induces μ. In this formulation, WMC corresponds to the process of calculating the value of $\mu_\nu(x)$ for some $x \in 2^{2^U}$ with a given definition of ν.

Relation to the classical (logic-based) view of WMC. Let $\mathcal{L}$ be a propositional logic with two atoms a and b as in Section 3 and $w\colon \{a, b, \neg a, \neg b\} \to \mathbb{R}_{\geq 0}$ a weight function defined as $w(a) = 0.3$, $w(\neg a) = 0.7$, $w(b) = 0.2$,

Name in logic	Boolean-algebraic notation	Set-theoretic notation
Atoms (elements of U)	a, b	a, b
Models (elements of 2^U)	$\neg a \wedge \neg b, a \wedge \neg b, \neg a \wedge b, a \wedge b$	$\emptyset, \{a\}, \{b\}, \{a, b\}$
Formulas (elements of 2^{2^U})	$\top$	$\{\emptyset, \{a\}, \{b\}, \{a, b\}\}$
	$\neg a \vee \neg b, a \to b$	$\{\emptyset, \{a\}, \{b\}\}, \{\emptyset, \{a\}, \{a, b\}\}$
	$b \to a, a \vee b$	$\{\emptyset, \{b\}, \{a, b\}\}, \{\{a\}, \{b\}, \{a, b\}\}$
	$\neg b, \neg a, a \leftrightarrow b$	$\{\emptyset, \{a\}\}, \{\emptyset, \{b\}\}, \{\emptyset, \{a, b\}\}$
	$(a \wedge \neg b) \vee (b \wedge \neg a), a, b$	$\{\{a\}, \{b\}\}, \{\{a\}, \{a, b\}\}, \{\{b\}, \{a, b\}\}$
	$\neg a \wedge \neg b, a \wedge \neg b, \neg a \wedge b, a \wedge b$	$\{\emptyset\}, \{\{a\}\}, \{\{b\}\}, \{\{a, b\}\}$
	$\bot$	$\emptyset$

Table 1: Notation for a logic with two atoms. The elements in both columns are listed in the same order.

$w(\neg b) = 0.8$. Furthermore, let Δ be a theory in $\mathcal{L}$ with a sole axiom a. Then Δ has two models: $\{a, b\}$ and $\{a, \neg b\}$ and its WMC [Chavira and Darwiche, 2008] is

$$\begin{aligned} \mathrm{WMC}(\Delta) &= \sum_{\omega \models \Delta} \prod_{\omega \models l} w(l) \\ &= w(a)w(b) + w(a)w(\neg b) = 0.3. \end{aligned} \tag{1}$$

Alternatively, we can define $\nu_a \colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ as $\nu_a(\{a\}) = 0.3$, $\nu_a(\emptyset) = 0.7$ and $\nu_b \colon 2^{\{b\}} \to \mathbb{R}_{\geq 0}$ as $\nu_b(\{b\}) = 0.2$, $\nu_b(\emptyset) = 0.8$. Let μ be the measure on 2^{2^U} induced by $\nu = \nu_a \cdot \nu_b$. Then, equivalently to Eq. (1), we can write

$$\begin{aligned} \mu(a) &= \nu(\{a, b\}) + \nu(\{a\}) \\ &= \nu_a(\{a\})\nu_b(\{b\}) + \nu_a(\{a\})\nu_b(\emptyset) = 0.3. \end{aligned}$$

Thus, one can equivalently think of WMC as summing over models of a theory or over atoms below an element of a Boolean algebra.

4.1 Not All Measures Are Factorable

Using this new definition of WMC, we can show that WMC with weights defined on literals is only able to capture a subset of all possible measures on a Boolean algebra. This can be demonstrated with a simple example.

Example 1. *Let $U = \{a, b\}$ be a set of atoms and $\mu \colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ a measure defined as $\mu(a \wedge b) = 0.72$, $\mu(a \wedge \neg b) = 0.18$, $\mu(\neg a \wedge b) = 0.07$, $\mu(\neg a \wedge \neg b) = 0.03$.[1] If μ could be represented using literal-weight (factored) WMC, we would have to find two weight functions $\nu_a \colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ and $\nu_b \colon 2^{\{b\}} \to \mathbb{R}_{\geq 0}$ such that $\nu = \nu_a \cdot \nu_b$ induces μ, i.e., ν_a and ν_b would have to satisfy this system of equations:*

$$\begin{aligned} \nu_a(\{a\}) \cdot \nu_b(\{b\}) &= 0.72 \\ \nu_a(\{a\}) \cdot \nu_b(\emptyset) &= 0.18 \\ \nu_a(\emptyset) \cdot \nu_b(\{b\}) &= 0.07 \\ \nu_a(\emptyset) \cdot \nu_b(\emptyset) &= 0.03, \end{aligned}$$

which has no solution.

Alternatively, we can let b depend on a and consider weight functions $\nu_a \colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ and $\nu_b \colon 2^{\{a,b\}} \to \mathbb{R}_{\geq 0}$ defined as $\nu_a(\{a\}) = 0.9$, $\nu_a(\emptyset) = 0.1$, and $\nu_b(\{a, b\}) = 0.8$, $\nu_b(\{a\}) = 0.2$, $\nu_b(\{b\}) = 0.7$, $\nu_b(\emptyset) = 0.3$. One can easily check that with these definitions ν indeed induces μ.

Note that in this case, we chose to interpret ν_b as $\Pr(b \mid a)$ while—with a different definition of ν_b that represents the joint probability distribution $\Pr(a, b)$—ν_b by itself could induce μ. In general, however, factorising the full weight function into several smaller functions often results in weight functions with smaller domains which leads to increased efficiency and decreased memory usage [Dudek *et al.*, 2020]. We can easily generalise this example further.

Theorem 2. *For any set U such that $|U| \geq 2$, there exists a non-factorable measure $2^{2^U} \to \mathbb{R}_{\geq 0}$.*

Since many measures of interest may not be factorable, a well-known way to encode them into instances of WMC is by adding more literals [Chavira and Darwiche, 2008]. We can use the measure-theoretic perspective on WMC to show that this is always possible, however, as ensuing sections will demonstrate, it often makes the inference task much harder in practice.[2]

Theorem 3. *For any set U and measure $\mu \colon 2^{2^U} \to \mathbb{R}_{\geq 0}$, there exists a set $V \supseteq U$, a factorable measure $\mu' \colon 2^{2^V} \to \mathbb{R}_{\geq 0}$, and a formula $f \in 2^{2^V}$ such that $\mu(x) = \mu'(x \wedge f)$ for all formulas $x \in 2^{2^U}$.*

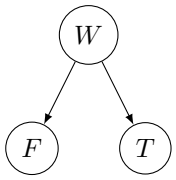
5 Encoding Bayesian Networks Using Conditional Weights

In this section, we describe a way to encode Bayesian networks into WMC without restricting oneself to factorable measures and thus having to add extra variables. We will refer to it as `cw`.

A Bayesian network is a directed acyclic graph with random variables as vertices that defines a probability distribution over them. Let $\mathcal{V}$ denote this set of random variables. For any random variable $X \in \mathcal{V}$, let $\operatorname{im} X$ denote its set of values and $\mathrm{pa}(X)$ its set of parents. The full probability distribution is then equal to $\prod_{X \in \mathcal{V}} \Pr(X \mid \mathrm{pa}(X))$. For discrete Bayesian networks (and we only consider discrete networks

[1] The value of μ on any other element of 2^{2^U} can be deduced from the definition of a measure.

[2] The proofs of this and other theoretical results can be found in the technical appendix.



w	$\Pr(W = w)$
1	0.5
0	0.5

w	f	$\Pr(F = f \mid W = w)$
1	1	0.6
1	0	0.4
0	1	0.1
0	0	0.9

w	t	$\Pr(T = t \mid W = w)$
1	l	0.2
1	m	0.4
1	h	0.4
0	l	0.6
0	m	0.3
0	h	0.1

Figure 1: An example Bayesian network with its CPTs

in this paper), each factor of this product can be represented by a CPT. See Figure 1 for an example Bayesian network that we will refer to throughout this section. For this network, $\mathcal{V} = \{W, F, T\}$, $\text{pa}(W) = \emptyset$, $\text{pa}(F) = \text{pa}(T) = \{W\}$, $\text{im}\, W = \text{im}\, F = \{0, 1\}$, and $\text{im}\, T = \{l, m, h\}$.

Definition 2 (Indicator variables)**.** *Let $X \in \mathcal{V}$ be a random variable. If X is binary (i.e., $|\text{im}\, X| = 2$), we can arbitrary identify one of the values as 1 and the other one as 0 (i.e, $\text{im}\, X \cong \{0, 1\}$). Then X can be represented by a single* indicator variable $\lambda_{X=1}$. *For notational simplicity, for any set S, we write $\lambda_{X=0} \in S$ or $S = \{\lambda_{X=0}, \dots\}$ to mean $\lambda_{X=1} \notin S$.*

On the other hand, if X is not binary, we represent X with $|\text{im}\, X|$ indicator variables, one for each value. We let

$$\mathcal{E}(X) = \begin{cases} \{\lambda_{X=1}\} & \text{if } |\text{im}\, X| = 2 \\ \{\lambda_{X=x} \mid x \in \text{im}\, X\} & \text{otherwise.} \end{cases}$$

denote the set of indicator variables for X and $\mathcal{E}^*(X) = \mathcal{E}(X) \cup \bigcup_{Y \in \text{pa}(X)} \mathcal{E}(Y)$ denote the set of indicator variables for X and its parents in the Bayesian network. Finally, let $U = \bigcup_{X \in \mathcal{V}} \mathcal{E}(X)$ denote the set of all indicator variables for all random variables in the Bayesian network.

For example, in the Bayesian network from Figure 1, $\mathcal{E}^*(T) = \{\lambda_{T=l}, \lambda_{T=m}, \lambda_{T=h}, \lambda_{W=1}\}$.

Algorithm 1 shows how a Bayesian network with vertices $\mathcal{V}$ can be represented as a weight function $\phi\colon 2^U \to \mathbb{R}_{\geq 0}$. The algorithm begins with the unit function and multiplies it by $\text{CPT}_X\colon 2^{\mathcal{E}^*(X)} \to \mathbb{R}_{\geq 0}$ for each random variable $X \in \mathcal{V}$. We call each such function a *conditional weight function* as it represents a conditional probability distribution. However, the distinction is primarily a semantic one: a function $2^{\{a,b\}} \to \mathbb{R}_{\geq 0}$ can represent $\Pr(a \mid b)$, $\Pr(b \mid a)$, or something else entirely, e.g., $\Pr(a \wedge b)$, $\Pr(a \vee b)$, etc.

For a binary random variable X, CPT_X is simply a sum of smaller functions, one for each row of the CPT. If X has more than two values, we also multiply CPT_X by 'clause' functions that restrict the value of $\phi(T)$ to zero whenever $|\mathcal{E}(X) \cap T| \neq 1$. For the Bayesian network in Figure 1, we get:

$$\begin{aligned} \text{CPT}_\text{F} &= 0.6[\lambda_{F=1}] \cdot [\lambda_{W=1}] + 0.4[\lambda_{F=0}] \cdot [\lambda_{W=1}] \\ &\quad + 0.1[\lambda_{F=1}] \cdot [\lambda_{W=0}] + 0.9[\lambda_{F=0}] \cdot [\lambda_{W=0}], \\ \text{CPT}_\text{T} &= ([\lambda_{T=l}] + [\lambda_{T=m}] + [\lambda_{T=h}]) \\ &\quad \cdot (\overline{[\lambda_{T=l}]} + \overline{[\lambda_{T=m}]}) \cdot (\overline{[\lambda_{T=l}]} + \overline{[\lambda_{T=h}]}) \\ &\quad \cdot (\overline{[\lambda_{T=m}]} + \overline{[\lambda_{T=h}]}) \cdot \dots. \end{aligned}$$

Algorithm 1: Encoding a Bayesian network

Data: vertices $\mathcal{V}$, probability distribution Pr
Result: $\phi\colon 2^U \to \mathbb{R}_{\geq 0}$
$\phi \leftarrow 1$;
for $X \in \mathcal{V}$ **do**
  let $\text{pa}(X) = \{Y_1, \dots, Y_n\}$;
  $\text{CPT}_X \leftarrow 0$;
  if $|\text{im}\, X| = 2$ **then**
    for $(y_i)_{i=1}^n \in \prod_{i=1}^n \text{im}\, Y_i$ **do**
      $p_1 \leftarrow \Pr(X = 1 \mid y_1, \dots, y_n)$;
      $p_0 \leftarrow \Pr(X \neq 1 \mid y_1, \dots, y_n)$;
      $\text{CPT}_X \leftarrow \text{CPT}_X$
        $+ p_1[\lambda_{X=1}] \cdot \prod_{i=1}^n [\lambda_{Y_i=y_i}]$
        $+ p_0\overline{[\lambda_{X=1}]} \cdot \prod_{i=1}^n [\lambda_{Y_i=y_i}]$;
  else
    let $\text{im}\, X = \{x_1, \dots, x_m\}$;
    for $x \in \text{im}\, X$ **and** $(y_i)_{i=1}^n \in \prod_{i=1}^n \text{im}\, Y_i$ **do**
      $p_x \leftarrow \Pr(X = x \mid y_1, \dots, y_n)$;
      $\text{CPT}_X \leftarrow \text{CPT}_X$
        $+ p_x[\lambda_{X=x}] \cdot \prod_{i=1}^n [\lambda_{Y_i=y_i}]$
        $+ \overline{[\lambda_{X=x}]} \cdot \prod_{i=1}^n [\lambda_{Y_i=y_i}]$;
    $\text{CPT}_X \leftarrow \text{CPT}_X \cdot (\sum_{i=1}^m [\lambda_{X=x_i}])$
      $\cdot \prod_{i=1}^m \prod_{j=i+1}^m (\overline{[\lambda_{X=x_i}]} + \overline{[\lambda_{X=x_j}]})$;
  $\phi \leftarrow \phi \cdot \text{CPT}_X$;
return ϕ;

5.1 Correctness

Algorithm 1 produces a function with a Boolean algebra as its domain. This function can be represented by an ADD [Bahar *et al.*, 1997]. The core of ADDMC works by taking an ADD $\psi\colon 2^U \to \mathbb{R}_{\geq 0}$ (expressed as a product of smaller ADDs) and returning $(\exists_U \psi)(\emptyset)$ [Dudek *et al.*, 2020]. In this section, we prove that the function ϕ produced by Algorithm 1 can be used by ADDMC to correctly compute any marginal probability of the Bayesian network that was encoded as ϕ.[3] We begin with Lemma 1 which shows that any conditional weight function produces the right answer when given a valid encoding of variable-value assignments relevant to the CPT.

Lemma 1. *Let $X \in \mathcal{V}$ be a random variable with parents $\text{pa}(X) = \{Y_1, \dots, Y_n\}$. Then $\text{CPT}_X\colon 2^{\mathcal{E}^*(X)} \to \mathbb{R}_{\geq 0}$ is*

[3] Note that it can just as well compute *any* probability expressed using the random variables in $\mathcal{V}$.

such that for any $x \in \operatorname{im} X$ *and* $(y_1, \ldots, y_n) \in \prod_{i=1}^{n} \operatorname{im} Y_i$,

$$\mathrm{CPT}_X(T) = \Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n),$$

where $T = \{\lambda_{X=x}\} \cup \{\lambda_{Y_i=y_i} \mid i = 1, \ldots, n\}$.

Now, Lemma 2 shows that ϕ represents the full probability distribution of the Bayesian network, i.e., it gives the right probabilities for the right inputs and zero otherwise.

Lemma 2. *Let* $\mathcal{V} = \{X_1, \ldots, X_n\}$. *Then*

$$\phi(T) = \begin{cases} \Pr(x_1, \ldots, x_n) & \text{if } T = \{\lambda_{X_i=x_i}\}_{i=1}^{n} \text{ for} \\ & \text{some } (x_i)_{i=1}^{n} \in \prod_{i=1}^{n} \operatorname{im} X_i \\ 0 & \text{otherwise,} \end{cases}$$

for all $T \in 2^U$.

We end with Theorem 4 that shows how ϕ can be combined with an encoding of a single variable-value assignment so that ADDMC would compute its marginal probability.

Theorem 4. *For any* $X \in \mathcal{V}$ *and* $x \in \operatorname{im} X$,

$$(\exists_U(\phi \cdot [\lambda_{X=x}]))(\emptyset) = \Pr(X = x).$$

5.2 Textual Representation

Algorithm 1 encodes a Bayesian network into a function on a Boolean algebra, but how does it relate to the standard interpretation of a WMC encoding as a formula in conjunctive normal form (CNF) together with a collection of weights? The factors of ϕ that restrict the values of indicator variables for non-binary random variables are already expressed as a product of sums of 0/1-valued functions, i.e., a kind of CNF. Disregarding these functions, each conditional weight function CPT_X is represented by a sum with a term for every subset of $\mathcal{E}^*(X)$. To encode these terms, we introduce *extended weight clauses* to the WMC format used by Cachet [Sang *et al.*, 2004]. For instance, here is a representation of the Bayesian network from Figure 1:

$\lambda_{T=l}$	$\lambda_{T=m}$	$\lambda_{T=h}$		0
	$-\lambda_{T=l}$	$-\lambda_{T=m}$		0
	$-\lambda_{T=l}$	$-\lambda_{T=h}$		0
	$-\lambda_{T=m}$	$-\lambda_{T=h}$		0
w	$\lambda_{W=1}$		0.5	0.5
w	$\lambda_{F=1}$	$\lambda_{W=1}$	0.6	0.4
w	$\lambda_{F=1}$	$-\lambda_{W=1}$	0.1	0.9
w	$\lambda_{T=l}$	$\lambda_{W=1}$	0.2	1
w	$\lambda_{T=m}$	$\lambda_{W=1}$	0.4	1
w	$\lambda_{T=h}$	$\lambda_{W=1}$	0.4	1
w	$\lambda_{T=l}$	$-\lambda_{W=1}$	0.6	1
w	$\lambda_{T=m}$	$-\lambda_{W=1}$	0.3	1
w	$\lambda_{T=h}$	$-\lambda_{W=1}$	0.1	1

where each indicator variable is eventually replaced with a unique positive integer. Each line prefixed with a w can be split into four parts: the 'main' variable (always not negated), conditions (possibly none), and two weights. For example, the line

$$w \quad \lambda_{T=m} \quad -\lambda_{W=1} \quad 0.3 \quad 1$$

encodes the function $0.3[\lambda_{T=m}] \cdot \overline{[\lambda_{W=1}]} + 1\overline{[\lambda_{T=m}]} \cdot \overline{[\lambda_{W=1}]}$ and can be interpreted as defining two conditional weights: $\nu(T = m \mid W = 0) = 0.3$, and $\nu(T \neq m \mid W = 0) = 1$, the former of which corresponds to a row in the CPT of T while the latter is artificially added as part of the encoding. In our encoding of Bayesian networks, it is always the case that, in each weight clause, either both weights sum to one, or the second weight is equal to one. Finally, note that the measure induced by these weights is not probabilistic (i.e., $\mu(\top) \neq 1$) by itself, but it becomes probabilistic when combined with the additional clauses that restrict what combinations of indicator variables can co-occur.

5.3 Changes to ADDMC

Here we describe two changes to ADDMC[4] [Dudek *et al.*, 2020] needed to adapt it to the new format.

First, ADDMC constructs the *Gaifman graph* of the input CNF formula as an aid for the algorithm's heuristics. This graph has as vertices the variables of the formula, and there is an edge between two variables u and v if there is a clause in the formula that contains both u and v. We extend this definition to functions on Boolean algebras, i.e., the factors of ϕ. For any pair of distinct variables $u, v \in U$, we draw an edge between them in the Gaifman graph if there is a function $\alpha\colon 2^X \to \mathbb{R}_{\geq 0}$ that is a factor of ϕ such that $u \in X$ and $v \in X$. For instance, a factor such as CPT_X will enable edges between all distinct pairs of variables in $\mathcal{E}^*(X)$.

Second, even though the function ϕ produced by Algorithm 1 is constructed to have 2^U as its domain, sometimes the domain is effectively reduced to 2^V for some $V \subset U$ by the ADD manipulation algorithms that optimise the ADD representation of a function. For a simple example, consider $\alpha : 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ defined as $\alpha(\{a\}) = \alpha(\emptyset) = 0.5$. Then α can be reduced to $\alpha'\colon 2^{\emptyset} \to \mathbb{R}_{\geq 0}$ defined as $\alpha'(\emptyset) = 0.5$. To compensate for these reductions, for the original WMC format with a weight function $w\colon U \cup \{\neg u \mid u \in U\} \to \mathbb{R}_{\geq 0}$, ADDMC would multiply its computed answer by $\prod_{u \in U \setminus V} w(u) + w(\neg u)$. With the new WMC format, we instead multiply the answer by $2^{|U \setminus V|}$. Each 'excluded' variable $u \in U \setminus V$ satisfies two properties: all weights associated with u are equal to 0.5 (otherwise the corresponding CPT would depend on u, and u would not be excluded), and all other CPTs are independent of u (or they may have a trivial dependence, where the probability stays the same if u is replaced with its complement). Thus, the CPT that corresponds to u still multiplies the weight of every atom in the Boolean algebra by 0.5, but the number of atoms under consideration is halved. To correct for this, we multiply the final answer by two for every $u \in U \setminus V$.

6 Experimental Comparison

In this section, we describe an experimental study comparing the five WMC encodings for Bayesian networks when run with ADDMC. The experiments were run on servers with Intel Xeon Gold 6138 and Intel Xeon E5-2630 processors[5] running Scientific Linux 7 with an 8 GiB memory limit and a 1000 s timeout.

[4] https://github.com/vardigroup/ADDMC

[5] Each instance is run on the same processor for all encodings.

Encoding	Unique	Fastest	Total
cd05	2	3	380
cd06	0	0	363
cw	234	935	980
d02	40	111	797
sbk05	12	44	765

Table 2: The numbers of instances (out of 1466) solved by ADDMC with each encoding (uniquely, faster than with others, and in total) under the described constraints

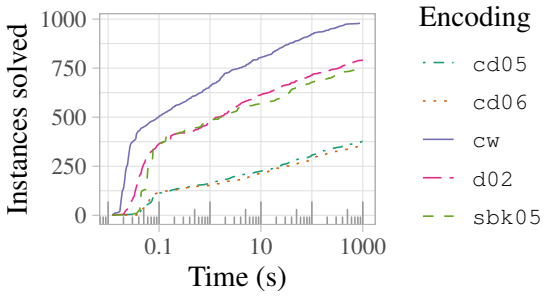


Figure 2: Cumulative number of instances solved by ADDMC over time using each encoding

Encoding(s)	Variables	Clauses/ADDs
cd05, cd06, sbk05	$\mathcal{O}(nv^{d+1})$	$\mathcal{O}(nv^{d+1})$
cw	$\mathcal{O}(nv)$	$\mathcal{O}(nv^2)$
d02	$\mathcal{O}(nv^{d+1})$	$\mathcal{O}(ndv^{d+1})$

Table 3: Asymptotic upper bounds on the numbers of variables and clauses/ADDs for each encoding

For each Bayesian network, we need to choose a probability to compute. Whenever a Bayesian network comes with an evidence file, we compute the probability of evidence. Otherwise, let X denote the last-mentioned vertex in the Bayesian network. If $\mathsf{true} \in \operatorname{im} X$, then we compute the marginal probability of $X = \mathsf{true}$. Otherwise, we pick the value of X which is listed first and calculate its marginal probability.

For all encodings other than cw, we use their implementation in Ace 3.0[6] with -encodeOnly and -noEclause flags. However, Ace was not used to encode evidence, as preliminary experiments revealed that the evidence-encoding implementation contains bugs that can lead to incorrect answers or a Java exception being thrown on some instances of the data set (and the source code is not publicly available). Instead, we simply list all the evidence as additional clauses in the encoding.

Note that both cd05 and cd06 purposefully produce overly relaxed encodings that contain extra models and thus yield incorrect probabilities [Chavira and Darwiche, 2005; Chavira and Darwiche, 2006]. These additional models are supposed to be filtered out during circuit compilation [Chavira and Darwiche, 2005], but this is not easily achievable with ADDMC. Nonetheless, we include both encodings in the experiments.

While encoding time itself was not measured because of different languages of implementation, it is worth noting that cw is linear in the total number of CPT rows and so is unlikely to be slower than, e.g., cd06, which relies on solving NP-complete problems as part of the encoding process [Chavira and Darwiche, 2006].

For experimental data, we use the Bayesian networks available with Ace and Cachet[7], most of which happen to be binary. We classify them into the following seven categories: • DQMR and • Grid networks as described by Sang *et al.* [2005], • Mastermind, and • Random Blocks from the work of Chavira *et al.* [2006], • remaining binary Bayesian networks that include Plan Recognition [Sang *et al.*, 2005], Friends and Smokers, Students and Professors [Chavira *et al.*, 2006], and tcc4f, and • non-binary classic Bayesian networks (alarm, diabetes, hailfinder, mildew, munin1–4, pathfinder, pigs, water). We run ADDMC with each of the five encodings once on each Bayesian network.

According to the cumulative plot in Figure 2, cw is consistently able to solve more instances with any given time limit than any other encoding, while the order of the remaining encodings from best to worst is exactly the opposite of that in previous literature (cf. Section 2). Specifically, cw solves the same number of instances in 8.144 s as d02 does in 1000 s, which is the second-best encoding in our experiments. Furthermore, Table 2 shows that cw was the fastest in 935 out of the 980 (i.e., 95.4 %) instances that it managed to solve. Finally, according to the scatter plots in Figure 3, cw is particularly promising on Grid networks while losing to d02 on some instances of the Mastermind data set. We conjecture that this is so either because the particular structure of the problem confuses the heuristics used by ADDMC or because these instances allow for significant simplifications that are exploited by d02 but not cw.

Explaining the performance benefits. Let $n = |\mathcal{V}|$ be the number of vertices in the Bayesian network, $d = \max_{X \in \mathcal{V}} |\mathrm{pa}(X)|$ the maximum in-degree (i.e., the number of parents), and $v = \max_{X \in \mathcal{V}} |\operatorname{im} X|$ the maximum number of values per variable. Table 3 shows how cw has fewer variables and fewer ADDs than any other encoding. However, note that these are upper bounds and most encodings (including cw) can be smaller in certain situations (e.g., with binary random variables or when a CPT has repeating probabilities). We equate clauses and ADDs (more specifically, factors of the function ϕ from Algorithm 1) here because ADDMC interprets each clause of any WMC encoding as a multiplicative factor of the ADD that represents the entire WMC instance [Dudek *et al.*, 2020]. For literal-weight encodings, each weight is also a factor, but that does not affect the asymptotic bounds in the table.

[6] http://reasoning.cs.ucla.edu/ace/

[7] https://www.cs.rochester.edu/u/kautz/Cachet/

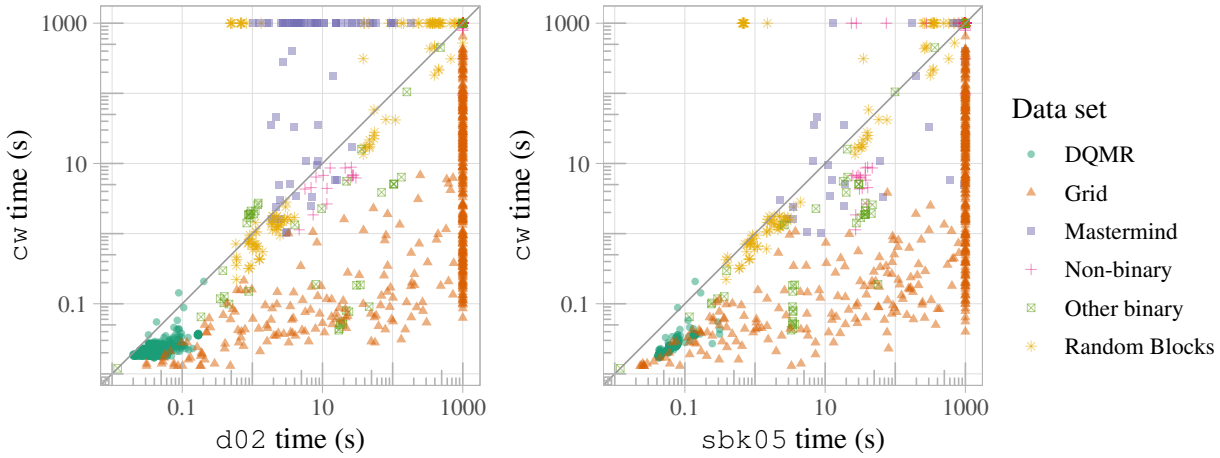


Figure 3: ADDMC inference time using `cw` compared to `d02` (left) and `sbk05` (right) for each instance in all data sets.

7 Discussion and Conclusion

WMC was originally motivated by an appeal to the success of SAT solvers in efficiently tackling an NP-complete problem [Sang *et al.*, 2005]. ADDMC does not rely on SAT-based algorithmic techniques, and our proposed format diverges even more from the DIMACS CNF format for Boolean formulas. To what extent are SAT-based methods still applicable? The answer depends significantly on the problem domain. For Bayesian networks, the rules describing that each random variable can only be associated with exactly one value were still encoded as clauses. As has been noted previously [Chavira and Darwiche, 2006], rows in CPTs with probabilities equal to zero or one can be represented as clauses as well. Therefore, our work can be seen as proposing a middle ground between #SAT and probabilistic inference.

While we chose to use ADDMC [Dudek *et al.*, 2020] as the WMC algorithm and Bayesian networks as a canonical example of a probabilistic inference task, these are only examples meant to illustrate the broader idea that choosing a more expressive representation of weights can outperform increasing the size of the problem to keep the weights simple. Indeed, in this work, we have provided a new theoretical perspective on the expressive power of WMC and illustrated the empirical benefits of that perspective. The same idea can be adapted to other inference problem domains such as probabilistic programs [Fierens *et al.*, 2015] as well as to search-based solvers such as Cachet [Sang *et al.*, 2004], although the extension may be less straightforward for other WMC techniques.

A Proofs

Theorem 1. *The function μ_ν is a measure.*

Proof. Note that $\mu_\nu(\bot) = 0$ since there are no atoms below $\bot$. Let $a, b \in 2^{2^U}$ be such that $a \wedge b = \bot$. By elementary properties of Boolean algebras, all atoms below $a \vee b$ are either below a or below b. Moreover, none of them can be below both a and b because then they would have to be below $a \wedge b = \bot$. Thus

$$\begin{aligned}\mu_\nu(a \vee b) &= \sum_{\{u\} \le a \vee b} \nu(u) = \sum_{\{u\} \le a} \nu(u) + \sum_{\{u\} \le b} \nu(u) \\ &= \mu_\nu(a) + \mu_\nu(b)\end{aligned}$$

as required. □

Theorem 3. *For any set U and measure $\mu\colon 2^{2^U} \to \mathbb{R}_{\ge 0}$, there exists a set $V \supseteq U$, a factorable measure $\mu'\colon 2^{2^V} \to \mathbb{R}_{\ge 0}$, and a formula $f \in 2^{2^V}$ such that $\mu(x) = \mu'(x \wedge f)$ for all formulas $x \in 2^{2^U}$.*

Proof. Let $V = U \cup \{f_m \mid m \in 2^U\}$, and $f = \bigwedge_{m \in 2^U} \{m\} \leftrightarrow f_m$. We define weight function $\nu\colon 2^V \to \mathbb{R}_{\ge 0}$ as $\nu = \prod_{v \in V} \nu_v$, where $\nu_v(\{v\}) = \mu(\{m\})$ if $v = f_m$ for some $m \in 2^U$ and $\nu_v(x) = 1$ for all other $v \in V$ and $x \in 2^{\{v\}}$. Let $\mu'\colon 2^{2^V} \to \mathbb{R}_{\ge 0}$ be the measure induced by ν. It is enough to show that μ and $x \mapsto \mu'(x \wedge f)$ agree on the atoms in 2^{2^U}. For any $\{a\} \in 2^{2^U}$,

$$\begin{aligned}\mu'(\{a\} \wedge f) &= \sum_{\{x\} \le \{a\} \wedge f} \nu(x) = \nu(a \cup \{f_a\}) \\ &= \nu_{f_a}(\{f_a\}) = \mu(\{a\})\end{aligned}$$

as required. □

Lemma 1. *Let $X \in \mathcal{V}$ be a random variable with parents $\mathrm{pa}(X) = \{Y_1, \ldots, Y_n\}$. Then $\mathrm{CPT}_X\colon 2^{\mathcal{E}^*(X)} \to \mathbb{R}_{\ge 0}$ is such that for any $x \in \operatorname{im} X$ and $(y_1, \ldots, y_n) \in \prod_{i=1}^n \operatorname{im} Y_i$,*

$$\mathrm{CPT}_X(T) = \Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n),$$

where $T = \{\lambda_{X=x}\} \cup \{\lambda_{Y_i = y_i} \mid i = 1, \ldots, n\}$.

Proof. If X is binary, then CPT_X is a sum of $2\prod_{i=1}^n |\operatorname{im} Y_i|$ terms, one for each possible assignment of values to variables $X, Y_1, \ldots, Y_n$. Exactly one of these terms is nonzero when applied to T, and it is equal to $\Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n)$ by definition.

If X is not binary, then $\left(\sum_{i=1}^{m}[\lambda_{X=x_i}]\right)(T) = 1$, and $\left(\prod_{i=1}^{m}\prod_{j=i+1}^{m}(\overline{[\lambda_{X=x_i}]} + \overline{[\lambda_{X=x_j}]})\right)(T) = 1$, so $\mathrm{CPT}_X(T) = \Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n)$ by a similar argument as before. □

Lemma 2. *Let $\mathcal{V} = \{X_1, \ldots, X_n\}$. Then*

$$\phi(T) = \begin{cases} \Pr(x_1, \ldots, x_n) & \text{if } T = \{\lambda_{X_i=x_i}\}_{i=1}^{n} \text{ for some } (x_i)_{i=1}^{n} \in \prod_{i=1}^{n} \operatorname{im} X_i \\ 0 & \text{otherwise,} \end{cases}$$

for all $T \in 2^U$.

Proof. If $T = \{\lambda_{X=v_X} \mid X \in \mathcal{V}\}$ for some $(v_X)_{X\in\mathcal{V}} \in \prod_{X\in\mathcal{V}} \operatorname{im} X$, then

$$\begin{aligned}\phi(T) &= \prod_{X\in\mathcal{V}} \Pr\left(X = v_X \,\middle|\, \bigwedge_{Y\in\mathrm{pa}(X)} Y = v_Y\right) \\ &= \Pr\left(\bigwedge_{X\in\mathcal{V}} X = v_X\right)\end{aligned}$$

by Lemma 1 and the definition of a Bayesian network. Otherwise there must be some non-binary random variable $X \in \mathcal{V}$ such that $|\mathcal{E}(X) \cap T| \neq 1$. If $\mathcal{E}(X) \cap T = \emptyset$, then $\left(\sum_{i=1}^{m}[\lambda_{X=x_i}]\right)(T) = 0$, and so $\mathrm{CPT}_X(T) = 0$, and $\phi(T) = 0$. If $|\mathcal{E}(X) \cap T| > 1$, then we must have two different values $x_1, x_2 \in \operatorname{im} X$ such that $\{\lambda_{X=x_1}, \lambda_{X=x_2}\} \subseteq T$ which means that $(\overline{[\lambda_{X=x_1}]} + \overline{[\lambda_{X=x_2}]})(T) = 0$, and so, again, $\mathrm{CPT}_X(T) = 0$, and $\phi(T) = 0$. □

Theorem 4. *For any $X \in \mathcal{V}$ and $x \in \operatorname{im} X$,*

$$(\exists_U(\phi \cdot [\lambda_{X=x}]))(\emptyset) = \Pr(X = x).$$

Proof. Let $\mathcal{V} = \{X, Y_1, \ldots, Y_n\}$. Then

$$\begin{aligned}(\exists_U(\phi \cdot [\lambda_{X=x}]))(\emptyset) &= \sum_{T\in 2^U} (\phi \cdot [\lambda_{X=x}])(T) \\ &= \sum_{\lambda_{X=x}\in T\in 2^U} \phi(T) \\ &= \sum_{\lambda_{X=x}\in T\in 2^U} \left(\prod_{Y\in\mathcal{V}} \mathrm{CPT}_Y\right)(T) \\ &= \sum_{(y_i)_{i=1}^{n}\in\prod_{i=1}^{n}\operatorname{im} Y_i} \Pr(x, y_1, \ldots, y_n) \\ &= \Pr(X = x)\end{aligned}$$

by:

- the proof of Theorem 1 by Dudek *et al.* [2020];
- if $\lambda_{X=x} \notin T \in 2^U$, then $(\phi \cdot [\lambda_{X=x}])(T) = \phi(T) \cdot [\lambda_{X=x}](T \cap \{\lambda_{X=x}\}) = \phi(T) \cdot 0 = 0$;
- Lemma 2;
- marginalisation of a probability distribution.

□

References

[Bahar *et al.*, 1997] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. *Formal Methods Syst. Des.*, 10(2/3):171–206, 1997.

[Chavira and Darwiche, 2005] Mark Chavira and Adnan Darwiche. Compiling Bayesian networks with local structure. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 1306–1312. Professional Book Center, 2005.

[Chavira and Darwiche, 2006] Mark Chavira and Adnan Darwiche. Encoding CNFs to empower component analysis. In Armin Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, volume 4121 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 2006.

[Chavira and Darwiche, 2008] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artif. Intell.*, 172(6-7):772–799, 2008.

[Chavira *et al.*, 2006] Mark Chavira, Adnan Darwiche, and Manfred Jaeger. Compiling relational Bayesian networks for exact inference. *Int. J. Approx. Reason.*, 42(1-2):4–20, 2006.

[Darwiche, 2001] Adnan Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *J. Appl. Non Class. Logics*, 11(1-2):11–34, 2001.

[Darwiche, 2002] Adnan Darwiche. A logical approach to factoring belief networks. In Dieter Fensel, Fausto Giunchiglia, Deborah L. McGuinness, and Mary-Anne Williams, editors, *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002*, pages 409–420. Morgan Kaufmann, 2002.

[Dudek *et al.*, 2020] Jeffrey M. Dudek, Vu Phan, and Moshe Y. Vardi. ADDMC: weighted model counting with algebraic decision diagrams. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1468–1476. AAAI Press, 2020.

[Fierens *et al.*, 2015] Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Sht. Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory Pract. Log. Program.*, 15(3):358–401, 2015.

[Kimmig *et al.*, 2017] Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt. Algebraic model counting. *J. Appl. Log.*, 22:46–62, 2017.

[Lagniez and Marquis, 2017] Jean-Marie Lagniez and Pierre Marquis. An improved decision-dnnf compiler. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 667–673. ijcai.org, 2017.

[Sang *et al.*, 2004] Tian Sang, Fahiem Bacchus, Paul Beame, Henry A. Kautz, and Toniann Pitassi. Combining component caching and clause learning for effective model counting. In *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings*, 2004.

[Sang *et al.*, 2005] Tian Sang, Paul Beame, and Henry A. Kautz. Performing Bayesian inference by weighted model counting. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 475–482. AAAI Press / The MIT Press, 2005.